IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Gerard Chauvel, et al.	§	Confirmation No.:	1887
		§		
Serial No.:	10/631,308	§		
		§	Group Art Unit:	2183
Filed:	July 31, 2003	§		
		§		
For:	Mixed Stack-Based RISC Processor	§	Examiner:	Petranek, Andrew J
		§		

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Atty Docket No.: TI-35433 (1962-05412)
Date: November 14, 2007

Sir:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal is filed concurrently herewith.

<u>TABLE OF CONTENTS</u>

I. REAL PARTY IN INTEREST

The real party in interest is Texas Instruments Inc., a Delaware corporation, having its principal place of business in Dallas, Texas. The Assignment from the inventors to Texas Instruments Inc. was recorded on July 31, 2003 at Reel/Frame 014362/0955.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-41.
Claim cancellations:	17, 28-29.
Added claims:	None.
Presently pending claims:	1-16, 18-27, 30-41.
Allowed claims:	None.
Presently appealed claims:	1-16, 18-27, 30-41.

IV. STATUS OF THE AMENDMENTS

No claims were amended after the non-final Office action dated January 22, 2007.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The specification is directed to mixed stack-based RISC processor.[1] At least some of the various embodiments are as in claim 1:

> A processor, comprising:
>
> a core;[2]
>
> a multi-entry stack[3] contained in said core and usable in at least a stack-based instruction set and comprising a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries at the top of a main stack[4] implemented in memory outside said core;[5]
>
> logic contained in said core and coupled to said stack, the logic manages the stack;[6]
>
> a plurality of registers[7] contained in said core and coupled to the logic[8] and addressable through a second instruction set that provides register-based and memory-based operations;[9]
>
> wherein said logic executes instructions from both said stack-based instruction set and said second instruction set;[10] and
>
> an instruction fetch logic contained in said core[11], said instruction fetch logic receives at least stack-based instructions from the stack-based instruction set[12].

Other illustrative embodiments are as in claim 8:

> A method of processing instructions in a processor, comprising:
>
> fetch logic in a core[13] of the processor receiving instructions from a first instruction set which comprises stack-based instructions;[14]

---

[1] Specification Title.

[2] Specification page 8, paragraph [0022], lines 1-2. The balance of this Appeal Brief uses a shorthand notation for citations to the Specification in the form: ([page],[paragraph number], [lines]). Thus, this illustrative citation in the shorthand form reads (3, [0016], lines 1-3). *See also*, Figures 2, element 120.

[3] (8, [0022], lines 4-5); Figure 2, element 146.

[4] (10, [0024], lines 2-8); Figure 2, element 146.

[5] (10, [0024], lines 2-8); Figure 2, element 122.

[6] (9, [0022], lines 5-7); Figure 2, element 152.

[7] (9, [0022], lines 2-4); Figure 2, element 140.

[8] (9, [0022], lines 2-4); Figure 2, element 152.

[9] (9, [0022], line 2-6); Figure 2, element 140.

[10] (11, [0026], lines 6-7); Figure 2, element 152.

[11] (11, [0026], lines 2-3); Figure 2, element 154.

[12] (9, [0022], lines 1-3); Figure 2, element 154.

[13] (11, [0026], lines 2-3); Figure 2, element 154.

> said fetch logic receiving instructions from a second instruction set which comprises memory-based and register-based instructions, and executing said received instructions from the first and second instruction sets in said core[15].

Yet still other illustrative embodiments are as in claim 12:

> A processor, comprising:
>
> a core;[16]
>
> a multi-entry stack contained in said core and having a top and usable in at least a stack-based instruction set;[17]
>
> logic contained in said core and coupled to said stack, the logic manages the stack[18], memory coupled to said logic and located outside said core;[19] and
>
> a plurality of registers[20] contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations;[21]
>
> wherein a first register includes an address through which the top of the stack is accessed and a second register in which a value at the top of the stack is stored;[22]
>
> wherein said multi-entry stack comprises a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries of a main stack implemented in said memory;[23]
>
> wherein said logic executes instructions from both said stack-based instruction set and said second instruction set[24]; and
>
> wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set[25].

---

[14] (11, [0026], lines 2-5); Figure 2, element 154.

[15] (11, [0026], lines 2-7); Figure 2, element 154 and 152.

[16] (3, [0016], lines 1-3); Figures 2, element 120.

[17] (10, [0024], lines 2-8); Figure 2, element 146.

[18] (9, [0022], lines 5-7); Figure 2, element 152.

[19] (10, [0024], lines 5-6); Figure 2, element 122.

[20] (9, [0022], lines 2-4); Figure 2, element 140.

[21] (9, [0022], line 2-6); Figure 2, element 140.

[22] (9, [0023], lines 8-12); Figure 3, elements R6 and R7.

[23] (10, [0024], lines 3-8); Figure 2, element 146.

[24] (11, [0026], lines 6-7); Figure 2, element 152.

[25] (12, [0029], lines 1-12); Figure 4, elements 142.

Other illustrative embodiments are as in claim 30:

> A system, comprising:
> a main processor unit;[26]
> a co-processor[27] having a core that comprises a hardware stack, fetch logic, and registers[28], said co-processor is coupled to the main processor unit, said fetch logic receiving stack-based instructions from a first instruction set[29], and the core of the co-processor is configured to execute the stack-based instructions and instructions from a second instruction set that provides memory-based and register-based operations;[30]
> wherein said hardware stack comprises a subset of entries at a top of a memory-based stack implemented in memory outside said core[31].

[26] (6, [0017], lines 2-4); Figure 1, element 104.

[27] (6, [0017], lines 2-4); Figure 1, element 102.

[28] (8, [0022], lines 1-6); Figure 2, elements 120, 146, 154 and 140.

[29] (11, [0026], lines 2-5); Figure 2, element 154.

[30] (11, [0026], lines 6-7); Figure 2, element 152.

[31] (10, [0024], lines 3-8); Figure 2, element 122.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-2, 4, and 6-7 are obvious under 35 USC § 103 over Feierbach et al. (U.S. Pat. No. 6,088,786) in view of Batten et al. (U.S. Pat. No. 6,256,725).

Whether claims 8-11 are anticipated under 35 USC §102(b) by Feierbach et al. (U.S. Pat. No. 6,088,786).

Whether claims 12-15, 18-24, and 27 are obvious under 35 USC § 103 over Feierbach et al. (U.S. Pat. No. 6,088,786) in view of Patel et al. (U.S. Pat. No. 6,826,749) in view of Batten et al. (U.S. Pat. No. 6,256,725) further in view of Hennessy et al. ("Computer Architecture: A Quantitative Approach").

Whether claims 30-35, 37, and 41 are obvious under 35 USC § 103 over Feierbach et al. (U.S. Pat. No. 6,088,786) in view of Hendler et al. (U.S. Pat. No. 6,473,777).

VII. ARGUMENT

A. Section §103 Rejections over Feierbach in view of Batten

1. Claims 1-2, 4, and 6-7

Claims 1-2, 4, and 6-7 stand rejected as allegedly anticipated by Feierbach. Claim 1 is representative of this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 U.S.C. § 282 shall apply to each of these claims individually.

Feierbach is directed towards coupling a stack based processor to a register based functional unit[32]. Feierbach teaches fetching stack instructions from a memory.

> In operation, stack and register processor 104 on client 102 fetches stack instructions from primary storage 110 through I/O interface[33].

The stack instructions fetched from the memory are decoded to determine whether the instructions are regular stack instructions or "extended stack instructions".

> After retrieving these instructions, stack and register processor 104 decodes the stack instructions and determines which are suitable for execution on a stack processor and which are suitable for execution one a register processor. In one embodiment, instructions involving multimedia…are classified as extended stack instructions… .If an instruction does not concern multimedia type operations, the instruction is classified as a regular stack instruction… .[34]

The extended stack instructions are stack instructions that are executed using registers by copying the stack values from a stack to a register.

> In response to these signals, copy-unit 206 copies data between a stack entry in stack 210 and register file 208… .[35]

> In response to control signals transmitted over register control lines 234, register processor 204 operates on the operands and values stored in register file 208 by copy unit 206.[36]

[32] Feierbach Title.

[33] Feierbach Col. 5, lines 62-63.

[34] Feierbach Col. 6, lines 63 to Col. 7, line 7.

[35] Feierbach Col. 8, lines 39-43.

[36] Feierbach Col. 9, lines 1-4.

Thus, Feierbach teaches that stack instructions can be executed on either a stack processor or a register processor using the operands and values of the stack.

Representative claim 1, by contrast specifically recites, "a multi-entry stack contained in said core and usable in at least **a stack-based instruction set** and comprising a plurality of entries…a plurality of registers contained in said core and coupled to the logic and addressable through **a second instruction set** that provides register-based and memory-based operations." Appellants respectfully submit that Feierbach and Batten fail to teach or fairly suggest such a processor. Feierbach teaches fetching a stack instructions and determining whether the instructions are regular stack instructions or extended stack instructions. Feierbach is silent as to a second instruction set that provides register-based and memory-based operations. Thus, even if hypothetically the teachings of Batten are precisely as the Office action suggests (which Appellants do not admit), the references still fail to teach or fairly suggest that a "a multi-entry stack contained in said core and usable in at least **a stack-based instruction set** and comprising a plurality of entries…. a plurality of registers contained in said core and coupled to the logic and addressable through **a second instruction set** that provides register-based and memory-based operations."

In the Response to Arguments section of the Office Action dated August 14, 2007, the Office Action states that "Feierbach disclosed a second instruction set, which are the extended instructions that are primarily executed on the register based processor."[37] The Appellants respectfully traverse. In accordance with Feierbach the extended instructions are one of two "types" of instructions that are part of **one** set of stack based instruction.

> Instruction decoder 222 includes an instruction preprocessor logic 227 to detect whether the instruction type is a regular stack instruction, suitable for execution on stack processor 202, or an extended stack instruction, suitable for execution on register processor 204.[38]

Thus, the Office Action's interpretation of the extended instructions as a second instruction set is inconsistent with the teachings of Feierbach.

[37] Office Action of August 14, 2007, page 24, last paragraph

[38] Feierbach Col. 7, lines 27-31.

Based on the foregoing, Appellants respectfully submit that the rejections of the claims of this grouping be overturned, and the claims set for issue.

2. Claim 3

Claim 3 stands rejected as allegedly obvious over Feierbach in view of Batten further in view of Patel. Claim 3 is allowable for at least the same reasons as delineated in Section VII(A)(1).

3. Claim 5

Claim 5 stands rejected as allegedly obvious over Feierbach in view of Batten further in view of Gee. Claim 5 is allowable for at least the same reasons as delineated in Section VII(A)(1).

B. Section § 102(b) Rejections over Feierbach.

1. Claims 8-11

Claims 8-11 stand rejected as allegedly anticipated by Feierbach. Claim 8 is representative of this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 U.S.C. § 282 shall apply to each of these claims individually.

Feierbach is directed towards coupling a stack based processor to a register based functional unit.[39] Specifically, Feierbach teaches that the processor fetches stack instructions, and decodes the fetched stack instructions to determine if they are regular stack instructions or extended stack instructions.[40] The extended stack instructions are stack instructions that are executed on registers by copying the stack values from a stack to a register.[41] Thus, Feierbach teaches that stack instructions can be executed on either a stack processor or a register processor using the operands and values of the stack.

Representative claim 8, by contrast, specifically recites, "fetch logic in a core of the processor receiving instructions from **a first instruction set** which comprises stack-

[39] Feierbach Title.

[40] Feierbach Col. 5, lines 62-67 to Col. 7, lines 1-7.

[41] Feierbach Col. 8, lines 39-43 and Feierbach Col. 9, lines 1-4.

based instructions; said fetch logic receiving instructions from **a second instruction set** which comprises memory-based and register-based instructions." Appellants respectfully submit that Feierbach fails to expressly or inherently teach suggest such a method. Feierbach teaches fetching a stack instructions and determining whether the instructions are regular stack instructions or extended stack instructions. Feierbach is silent as to a second instruction set that provides register-based and memory-based operations. Thus, the Appellants respectfully submit that Feierbach fails to expressly or inherently teach "fetch logic in a core of the processor receiving instructions from **a first instruction set** which comprises stack-based instructions; said fetch logic receiving instructions from **a second instruction set** which comprises memory-based and register-based instructions."

In Response to Arguments section, the Office Action states that "element 227 fetches all instructions…defines a second instruction set, extended instructions comprising mainly multimedia operations, are primarily executed on the register based processor."[42] The Appellants respectfully traverse. In accordance with Feierbach the extended instructions are one of two "types" of instructions that are part of **one** set of stack based instruction.

> Instruction decoder 222 includes an instruction preprocessor logic 227 to detect whether the instruction type is a regular stack instruction, suitable for execution on stack processor 202, or an extended stack instruction, suitable for execution on register processor 204.[43]

Thus, the Office Action's interpretation of the extended instructions as a second instruction set is inconsistent with the teachings of Feierbach.

Based on the foregoing, Appellants respectfully submit that the rejections of the claims in this grouping be reversed, and the claims set for issue.

C. Section § 103 Rejections over Feierbach in view of Patel in view of Batten further in view of Hennessy

1. Claims 12-15, 18-24, and 27

Claims 12-15, 18-24 and 27 stand rejected as allegedly obvious over Feierbach in view of Patel in view of Batten further in view of Hennessy. Claim 12 is representative of

[42] Office Action of August 14, 2007, page 25, second paragraph.

[43] Feierbach Col. 7, lines 27-31.

this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 U.S.C. § 282 shall apply to each of these claims individually.

Feierbach is directed towards coupling a stack based processor to a register based functional unit.[44] Specifically, Feierbach teaches that the processor fetches stack instructions, and decodes the fetched stack instructions to determine if they are regular stack instructions or extended stack instructions.[45] The extended stack instructions are stack instructions that are executed on registers by copying the stack values from a stack to a register.[46] Thus, Feierbach teaches that stack instructions can be executed on either a stack processor or a register processor using the operands and values of the stack.

Representative claim 12, specifically recites, "a multi-entry stack contained in said core and having a top and usable in at least **a stack-based instruction set**...a plurality of registers contained in said core and coupled to the logic and addressable through **a second instruction set** that provides register-based and memory-based operations". Appellants respectfully submit that Feierbach and Patel and Batten and Hennessy fail to teach or fairly suggest such a processor. Feierbach teaches fetching a stack instructions and determining whether the instructions are regular stack instructions or extended stack instructions. Feierbach is silent as to a second instruction set that provides register-based and memory-based operations. Thus, even if hypothetically the teachings of Patel and Batten and Hennessy are precisely as the Office action suggests (which Appellants do not admit), the references still fail to teach or fairly suggest that a "a multi-entry stack contained in said core and having a top and usable in at least **a stack-based instruction set**...a plurality of registers contained in said core and coupled to the logic and addressable through **a second instruction set** that provides register-based and memory-based operations."

[44] Feierbach Title.

[45] Feierbach Col. 5, lines 62-67 to Col. 7, lines 1-7.

[46] Feierbach Col. 8, lines 39-43 and Feierbach Col. 9, lines 1-4.

Based on the foregoing, Appellants respectfully submit that the rejections of the claims of this grouping be overturned, and the claims set for issue.

2. Claims 16 and 26

Claims 16 and 26 stand rejected as allegedly obvious over Feierbach in view of Patel in view of Batten in view of Hennessy further in view of Gee. Claim 16 and 26 are allowable for at least the same reasons as delineated in Section VII(C)(1).

3. Claim 25

Claim 25 stands rejected as allegedly obvious over Feierbach in view of Patel in view of Batten in view of Hennessy in view Hendler further in view of Brassac. Claim 25 is allowable for at least the same reasons as delineated in Section VII(C)(1).

D. Section § 103 Rejections over Feierbach in view of Hendler

1. Claims 30-35, 37, and 41

Claims 30-35, 37, and 41 stand rejected as allegedly obvious over Feierbach in view of Hendler. Claim 30 is representative of this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 U.S.C. § 282 shall apply to each of these claims individually.

Feierbach is directed towards coupling a stack based processor to a register based functional unit.[47] Specifically, Feierbach teaches that the processor fetches stack instructions, and decodes the fetched stack instructions to determine if they are regular stack instructions or extended stack instructions.[48] The extended stack instructions are stack instructions that are executed on registers by copying the stack values from a stack to a register.[49] Thus, Feierbach teaches that stack instructions can be executed on either a stack processor or a register processor using the operands and values of the stack.

Representative claim 30, specifically recites, "a co-processor having a core that comprises a hardware stack, fetch logic, and registers, said co-processor is coupled to the main processor unit, said fetch logic receiving stack-based instructions from **a first**

[47] Feierbach Title.

[48] Feierbach Col. 5, lines 62-67 to Col. 7, lines 1-7.

[49] Feierbach Col. 8, lines 39-43 and Feierbach Col. 9, lines 1-4.

instruction set, and the core of the co-processor is configured to execute the stack-based instructions and instructions from **a second instruction set** that provides memory-based and register-based operations". Appellants respectfully submit that Feierbach and Hendler fail to teach or fairly suggest such a processor. Feierbach teaches fetching a stack instructions and determining whether the instructions are regular stack instructions or extended stack instructions. Feierbach is silent as to a second instruction set that provides register-based and memory-based operations. Thus, even if hypothetically the teachings of Hendler are precisely as the Office action suggests (which Appellants do not admit), the references still fail to teach or fairly suggest that a "a co-processor having a core that comprises a hardware stack, fetch logic, and registers, said co-processor is coupled to the main processor unit, said fetch logic receiving stack-based instructions from **a first instruction set**, and the core of the co-processor is configured to execute the stack-based instructions and instructions from **a second instruction set** that provides memory-based and register-based operations."

Based on the foregoing, Appellants respectfully submit that the rejections of the claims of this grouping be overturned, and the claims set for issue.

2. Claim 36

Claim 36 stands rejected as allegedly obvious over Feierbach in view of Hendler further in view of Patel. Claim 36 is allowable for at least the same reasons as delineated in Section VII(D)(1).

3. Claims 38-40

Claims 38-40 stand rejected as allegedly obvious over Feierbach in view of Hendler further in view of Gee. Claims 38-40 are allowable for at least the same reasons as delineated in Section VII(D)(1).

E. Conclusion

For the reasons stated above, Appellants respectfully submit that the Examiner erred in rejecting all pending claims. It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R.

§ 1.136(a), and any fees required (including fees for net addition of claims) are hereby authorized to be charged to the Texas Instruments, Inc. Deposit Account No. 20-0668.

Respectfully submitted,

/Utpal D. Shah/

Utpal D. Shah
PTO Reg. No. 60,047
CONLEY ROSE, P.C.
(512) 610-3440 (Phone)
(512) 610-3456 (Fax)
AGENT FOR APPELLANTS

VIII. CLAIMS APPENDIX

1. (Previously Presented) A processor, comprising:

a core;

a multi-entry stack contained in said core and usable in at least a stack-based instruction set and comprising a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries at the top of a main stack implemented in memory outside said core;

logic contained in said core and coupled to said stack, the logic manages the stack; and

a plurality of registers contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations;

wherein said logic executes instructions from both said stack-based instruction set and said second instruction set; and

an instruction fetch logic contained in said core, said instruction fetch logic receives at least stack-based instructions from the stack-based instruction set.

2. (Previously Presented) The processor of claim 1 wherein the multi-entry stack has a top and the stack is accessible within the second instruction set through at least one of the registers in which a value is stored that is present at the top of the multi-entry stack.

3. (Previously Presented) The processor of claim 1 wherein the multi-entry stack has a top that is addressable by a memory mapped address, and the memory mapped address is stored in a register which is accessed by the second instruction set.

4. (Previously Presented) The processor of claim 1 wherein the stack-based instruction set accesses operands from the multi-entry stack and places results from operations on the multi-entry stack and, as a result of accessing operands from the multi-entry stack and placing results on the multi-entry stack, at least some of the registers are updated.

5. (Original) The processor of claim 1 further comprising a first program counter usable in the execution of the stack-based instruction set and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the stack-based and second instruction sets.

6. (Previously Presented) The processor of claim 1 further comprising a pair of parallel address generation units coupled to said logic which are used to compute memory source and destination addresses and wherein a register includes the top of the multi-entry stack, thereby permitting a block of data to be moved between a memory area and the multi-entry stack by execution of a single instruction with a repeat loop.

7. (Original) The processor of claim 1 wherein the second instruction set comprises an instruction that retrieves operands from memory, performs a computation on the operands, and places the result on the stack.

8. (Previously Presented) A method of processing instructions in a processor, comprising:

fetch logic in a core of the processor receiving instructions from a first instruction set which comprises stack-based instructions;

said fetch logic receiving instructions from a second instruction set which comprises memory-based and register-based instructions; and

executing said received instructions from the first and second instruction sets in said core.

9. (Original) The method of claim 8 further comprising forming a sequence of instructions from both of said first and second instruction sets.

10. (Previously Presented) The method of claim 8 further comprising executing an instruction from said second instruction set that targets a stack included in said core, said stack having a top, and storing a value at the top of the stack in a register in the processor.

11. (Original) The method of claim 10 further comprising updating an address stored in another register that points to the top of the stack.

12. (Previously Presented) A processor, comprising:

a core;

a multi-entry stack contained in said core and having a top and usable in at least a stack-based instruction set;

logic contained in said core and coupled to said stack, the logic manages the stack;

memory coupled to said logic and located outside said core; and

a plurality of registers contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations;

wherein a first register includes an address through which the top of the stack is accessed and a second register in which a value at the top of the stack is stored;

wherein said multi-entry stack comprises a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries of a main stack implemented in said memory;

wherein said logic executes instructions from both said stack-based instruction set and said second instruction set; and

wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set.

13. (Previously Presented) The processor of claim 12 wherein the stack-based instruction set accesses operands from the multi-entry stack and places results from operations on the multi-entry stack and, as a result of accessing operands from the multi-entry stack and placing results on the multi-entry stack thereby causing the address in the first register to be changed.

14. (Original) The processor of claim 13 wherein the address in the first register is incremented or decremented depending on whether the register is used as a source or a destination, respectively, for an operation.

15. (Original) The processor of claim 12 wherein the stack-based instruction set comprises Java Bytecodes.

16. (Original) The processor of claim 12 further comprising a first program counter usable in the execution of the first instruction set and a second program counter usable in the execution of code that comprises instructions from both the first and second instruction sets.

17. (Canceled)

18. (Previously presented) The processor of claim 12 wherein at least one of the registers includes an offset usable in the calculation of addresses.

19. (Previously Presented) The processor of claim 12 wherein the second instruction set comprises an instruction that moves data from a register or memory to a register, and consequently to the multi-entry stack.

20. (Original) The processor of claim 19 wherein the instruction that moves data includes a plurality of bits of that encode one of a plurality of addressing modes.

21. (Original) The processor of claim 20 wherein the addressing modes include a mode in which the instruction that moves data includes an immediate value and a reference to a register containing a base address, wherein the immediate value and the base address are added together to generate a source memory address for the move instruction.

22. (Original) The processor of claim 20 wherein the addressing modes include a mode in which the instruction that moves data includes a reference to register in which a source

memory address is stored to be used in the move instruction, and the source memory address in the referenced register is incremented by an immediate value also included in the move instruction.

23. (Original) The processor of claim 20 wherein the addressing modes include a mode in which the instruction that moves data includes references to two registers in which memory addresses are stored, one register being a predetermined index register, the memory addresses from the two registers are added together to calculate a source memory address used to complete the move instruction, and the address in the predetermined index register is incremented.

24. (Original) The processor of claim 20 wherein the addressing modes include a mode in which the instruction that moves data includes references to two registers in which memory addresses are stored, the memory addresses are added together to calculate the memory address used to complete the move instruction.

25. (Original) The processor of claim 12 wherein the processor is configured to be coupled to a separate processor on which an operating system is executed.

26. (Original) The processor of claim 12 further comprising a first program counter usable in the execution of the stack-based instruction set and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the stack-based and second instruction sets.

27. (Previously Presented) The processor of claim 12 further comprising a pair of parallel address generation units coupled to said logic which are used to compute memory source and destination addresses and wherein a register includes the top of the multi-entry stack, thereby permitting a block of data to be moved between a memory area and the multi-entry stack by execution of a single instruction with a repeat loop.

28-29. (Canceled)

30. (Previously Presented) A system, comprising:

a main processor unit;

a co-processor having a core that comprises a hardware stack, fetch logic, and registers, said co-processor is coupled to the main processor unit, said fetch logic receiving stack-based instructions from a first instruction set, and the core of the co-processor is configured to execute the stack-based instructions and instructions from a second instruction set that provides memory-based and register-based operations;

wherein said hardware stack comprises a subset of entries at a top of a memory-based stack implemented in memory outside said core.

31. (Original) The system of claim 30 wherein the stack-based instructions comprise Java bytecodes.

32. (Original) The system of claim 31 further including a compiler coupled to said co-processor, said compiler receives Java bytecodes and replaces at least one group of bytecodes by a sequence of instructions from the second instruction set and provides said sequence to the co-processor for execution.

33. (Original) The system of claim 32 wherein the sequence also includes stack-based instructions from the first instruction set.

34. (Original) The system of claim 30 wherein the system comprises a cellular telephone.

35. (Previously Presented) The system of claim 30 wherein the hardware stack has a top and is accessible within the second instruction set through at least one of the registers in which a value is stored that is present at the top of the hardware stack.

36. (Previously Presented) The system of claim 30 wherein the top of the hardware stack is addressable by a memory mapped address, and the memory mapped address is stored in a register which is accessed by the second instruction set.

37. (Previously Presented) The system of claim 30 wherein the stack-based instruction set accesses operands from the hardware stack and places results from operations on the hardware stack and, as a result of accessing operands from the hardware stack and placing results on the hardware stack, at least some of the registers are updated.

38. (Original) The system of claim 30 further comprising a first program counter usable in the execution of the stack-based instruction set and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the stack-based and second instruction sets.

39. (Original) The system of claim 38 wherein the first and second program counters are stored in said registers.

40. (Original) The system of claim 30 wherein the co-processor further comprises a first program counter usable in the execution of the first instruction set and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the first and second instruction sets.

41. (Previously Presented) The system of claim 30 further comprising a memory area and wherein the co-processor further comprises a pair of parallel address generation units coupled to said logic which are used to compute memory source and destination addresses and wherein a register includes the top of the hardware stack, thereby permitting a block of data to be moved between a memory area and the stack by execution of a single instruction with a repeat loop.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.